Лекция 15. Анализ ряда с помощью языка Python в зависимости от времени

На работе я практически ежедневно сталкиваюсь с теми или иными задачами, связанными с временными рядам. Чаще всего возникает вопрос — а что у нас будет происходить с нашими показателями в ближайший день/неделю/месяц/пр. — сколько игроков установят приложения, сколько будет онлайна, как много действий совершат пользователи, и так далее. К задаче прогнозирования можно подходить по-разному, в зависимости от того, какого качества должен быть прогноз, на какой период мы хотим его строить, и, конечно, как долго нужно подбирать и настраивать параметры модели для его получения.

Начнем с простых методов анализа и прогнозирования — скользящих средних, сглаживаний и их вариаций.

Движемся, сглаживаем и оцениваем

Временной ряд – это последовательность значений, описывающих протекающий во времени процесс, измеренных в последовательные моменты времени, обычно через равные промежутки

Таким образом, данные оказываются упорядочены относительно неслучайных моментов времени, и, значит, в отличие от случайных выборок, могут содержать в себе дополнительную информацию, которую мы постараемся извлечь.

Импортируем нужные библиотеки. В основном нам понадобится модуль statsmodels, в котором реализованы многочисленные методы статистического моделирования, в том числе для временных рядов. Для поклонников R, пересевших на питон, он может показаться очень родным, так как поддерживает написание формулировок моделей в стиле 'Wage ~ Age + Education'.

```
import sys

import warnings

warnings.filterwarnings('ignore')

from tqdm import tqdm

import pandas as pd

import numpy as np

from sklearn.metrics import mean_absolute_error, mean_squared_error

import statsmodels.formula.api as smf

import statsmodels.tsa.api as smt

import statsmodels.api as sm

import scipy.stats as scs

from scipy.optimize import minimize
```

```
import matplotlib.pyplot as plt

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
from plotly import graph_objs as go
init_notebook_mode(connected = True)

def plotly_df(df, title = ''):
    data = []

    for column in df.columns:
        trace = go.Scatter(
            x = df.index,
            y = df[column],
            mode = 'lines',
            name = column
        )
        data.append(trace)

    layout = dict(title = title)
    fig = dict(data = data, layout = layout)
    iplot(fig, show_link=False)

dataset = pd.read_csv('hour_online.csv', index_col=['Time'], parse_dates=['Time'])
plotly_df(dataset, title = "Online users")
```

Rolling window estimations

Начнем моделирование с наивного предположения — "завтра будет, как вчера", но вместо модели вида y^t=yt−1 будем считать, что будущее значение переменной зависит от среднего n её предыдущих значений, а значит, воспользуемся **скользящей средней**.

$$\hat{y}_{t} = \frac{1}{k} \displaystyle\sum^{k-1}_{n=0} y_{t-n}$$

Реализуем эту же функцию в питоне и посмотрим на прогноз, построенный по последнему наблюдаемому дню (24 часа)

```
def moving_average(series, n): return np.average(series[-n:]) moving_average(dataset.Users, 24)
```

```
Out: 29858.333333333332
```

К сожалению, такой прогноз долгосрочным сделать не удастся — для получения предсказания на шаг вперед предыдущее значение должно быть фактически наблюдаемой величиной. Зато у скользящей средней есть другое применение — сглаживание исходного ряда для выявления трендов. В пандасе есть готовая реализация — DataFrame.rolling(window).mean(). Чем больше зададим ширину интервала — тем более сглаженным окажется тренд. В случае, если данные сильно зашумлены, что особенно часто встречается, например, в финансовых показателях, такая процедура может помочь с определением общих паттернов.

Для нашего ряда тренды и так вполне очевидны, но если сгладить по дням, становится лучше видна динамика онлайна по будням и выходным (выходные — время поиграть), а

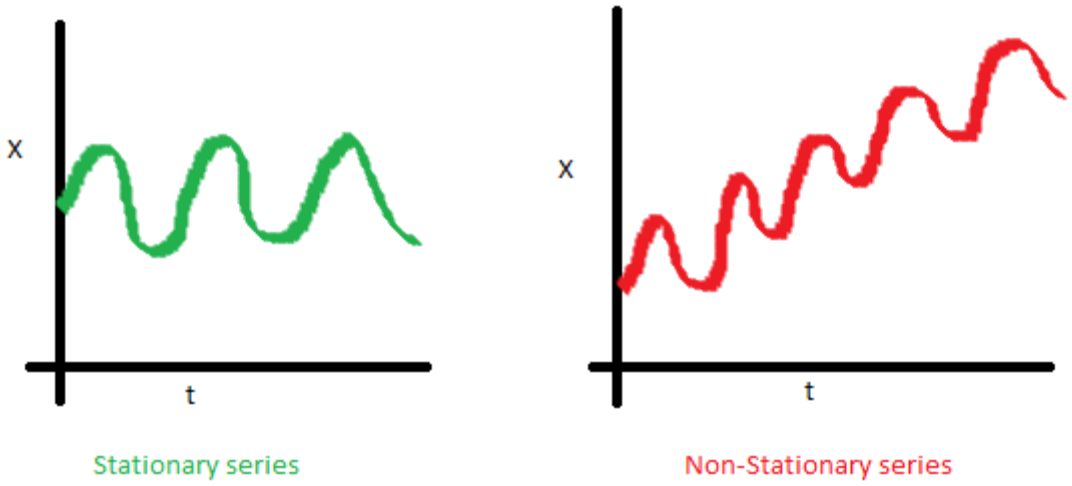
недельное сглаживание хорошо отражает общие изменения, связанные с резким ростом числа активных игроков в феврале и последующим снижением в марте.
Модификацией простой скользящей средней является взвешенная средняя, внутри которой наблюдениям придаются различные веса, в сумме дающие единицу, при этом обычно последним наблюдениям присваивается больший вес.

```
\hat{y}_{t} = \displaystyle\sum^{k}_{n=1} \omega_n y_{t+1-n}
```

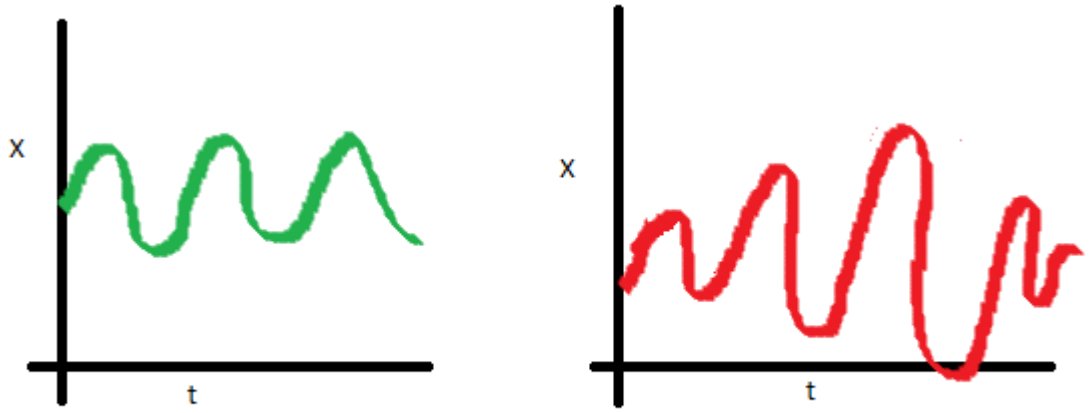
Стационарность, единичные корни

Перед тем, как перейти к моделированию, стоит сказать о таком важном свойстве временного ряда, как **стационарность**.
Под стационарностью понимают свойство процесса не менять своих статистических характеристик с течением времени, а именно постоянство матожидания, постоянство дисперсии (она же [гомоскедастичность]) и независимость ковариационной функции от времени (должна зависеть только от расстояния между наблюдениями). Наглядно можно посмотреть на эти свойства на картинках, взятых из поста [Sean Abu]:

- Временной ряд справа не является стационарным, так как его матожидание со временем растёт

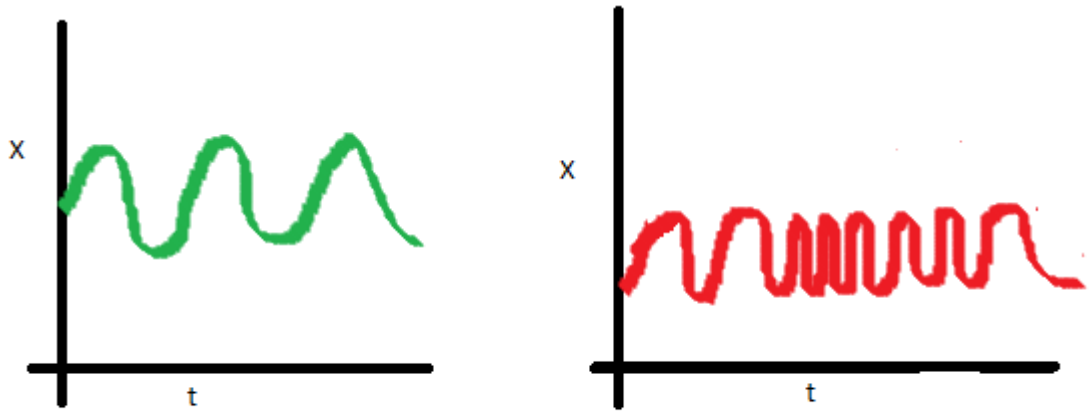


- Здесь не повезло с дисперсией — разброс значений ряда существенно варьируется в зависимости от периода



Stationary series Non-Stationary series

- Наконец, на последнем графике видно, что значения ряда внезапно становятся ближе друг ко другу, образуя некоторый кластер, а в результате получаем непостоянство ковариаций



Stationary series Non-Stationary series

Почему стационарность так важна? По стационарному ряду просто строить прогноз, так как мы полагаем, что его будущие статистические характеристики не будут отличаться от наблюдаемых текущих. Большинство моделей временных рядов так или иначе моделируют и предсказывают эти характеристики (например, матожидание или дисперсию), поэтому в случае нестационарности исходного ряда предсказания окажутся неверными. К сожалению, большинство временных рядов, с которыми приходится сталкиваться за пределыми учебных материалов, стационарными не являются, но с этим можно (и нужно) бороться.

Чтобы бороться с нестационарностью, нужно узнать её в лицо, потому посмотрим, как её детектировать. Для этого обратимся к белому шуму и случайному блужданию, чтобы выяснить как попасть из одного в другое бесплатно и без смс.

Избавляемся от нестационарности и строим SARIMA

Попробуем теперь построить ARIMA модель для онлайна игроков, пройдя все ~~круги ада~~ стадии приведения ряда к стационарному виду. Про саму модель уже не раз писали на

хабре — Построение модели SARIMA с помощью Python+R, Анализ временных рядов с помощью python, поэтому подробно останавливаться на ней не буду.